Interacting with Computers xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Interacting with Computers

journal homepage: www.elsevier.com/locate/intcom

Setting the stage – Embodied and spatial dimensions in emerging programming practices

Martin Jonsson [a], Jakob Tholander [b], Ylva Fernaeus [b,*]

[a] Södertörn University, School of Communication, Technology and Design, SE-141 89 Huddinge, Sweden
[b] Stockholm University, Swedish Institute of Computer Science, Forum 100, SE-164 40 Kista, Sweden

ARTICLE INFO

Article history:
Received 14 October 2008
Accepted 14 October 2008
Available online xxxx

Keywords:
Interaction design
Embodied interaction
Physical user interfaces
Embodied performance
Programming practice

ABSTRACT

In the design of interactive systems, developers sometimes need to engage in various ways of physical performance in order to communicate ideas and to test out properties of the system to be realised. External resources such as sketches, as well as bodily action, often play important parts in such processes, and several methods and tools that explicitly address such aspects of interaction design have recently been developed. This combined with the growing range of pervasive, ubiquitous, and tangible technologies add up to a complex web of physicality within the practice of designing interactive systems. We illustrate this dimension of systems development through three cases which in different ways address the design of systems where embodied performance is important. The first case shows how building a physical sport simulator emphasises a shift in activity between programming and debugging. The second case shows a build-once run-once scenario, where the fine-tuning and control of the run-time activity gets turned into an act of in situ performance by the programmers. The third example illustrates the explorative and experiential nature of programming and debugging systems for specialised and autonomous interaction devices. This multitude in approaches in existing programming settings reveals an expanded perspective of what practices of interaction design consist of, emphasising the interlinking between design, programming, and performance with the system that is being developed.

1. Introduction

The area of human–computer interaction has increasingly recognised the physical and material dimensions of acting with and around computational artefacts. This is mirrored through the central theme of much contemporary research concerned with understanding the social and physical interdependencies in relation to people's interaction with technology. This has most prominently been conceptualised through notions of embodied interaction and in studies of situated conduct, e.g., by work of Dourish (2001) and Heath and Luff (2000). Notions of physicality and embodiment are also central in research on pervasive computing and tangible interfaces, in which the particular physical manifestation of a computational artefact and its consequences for people's interaction with and through the artefact is commonly brought to discussion (see Hornecker and Buur, 2006; Klemmer et al., 2006).

The premise of this paper is that this development involves a number of new challenges for the designers and programmers of interactive artefacts and systems. This is partly since interactive systems increasingly involve artefacts that support bodily and social performance in physical spaces, but also as the tools that developers use have developed from text-based, through graphical, towards systems based on multimodal, and different forms of physical interaction. Early programming resources did, for instance, take the form of holes in physical punch cards, while more recent formats include textual codes, graphical rewrite rules (Canfield Smith et al., 2001), or even animated (Kahn, 1996) and tangible (Horn and Jacob, 2007) program representations. Currently, a range of physical and tangible resources for various aspects of programming has been developed, including tangible forms for representing programming constructs (McNerny, 2004), physical tools for construction of code structures displayed on a computer screen (Fernaeus and Tholander, 2006a), and methods of tracking and recording physical manipulation of actuated devices (Frei et al., 2000; Raffle et al., 2004; Hartmann et al., 2007). There is also a generally increasing focus on social and situated aspects of system development, manifested, for example, in the form of resources for collaboration such as version management software, open-source tools and public class libraries.

In our earlier work, we have explored how physical and visual programming tools contribute to a reshaping of the activity of programming, from intellectual and individual, towards getting the character of a largely social activity relying heavily on physical action and material resources. To describe this shift, we studied so-

* Corresponding author. Tel.: +46 708763301.
E-mail addresses: martin.jonsson@sh.se (M. Jonsson), jakob.tholander@sh.se (J. Tholander), ylva@sics.se (Y. Fernaeus).

0953-5438/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.intcom.2008.10.004

cial performances and their role in programming activity (Fernaeus and Tholander, 2006a), emphasising how visual and tangible forms of representations allow people to involve bodily actions such as pointing and gesture in a more direct sense than is possible with traditional text-based or symbolic programming representations. In analyses of children making animated games together on a PC, a central aspect of the negotiation did, for instance, concern how objects in the final game should move and behave. To discuss such aspects, the children made extensive use of gestures, and also external resources such as sketches on a piece of paper. The dynamic and spatial properties of the final application thus presented a number of challenges that the participants addressed through bodily performance within the programming activity. Similarly, extensive use of bodily action has been observed among adults building administrative software systems when talking about less tangible aspects of their designs (Tholander et al., 2008, see Fig. 1).

In this paper, we further explore how spatial and physical aspects, not of the tools but of the *intended interactive setting* affect the character of a programming activity. This further emphasises the multitude of understandings of what programming is and what it should be. We focus specifically on the activity of building interactive systems that have physicality and spatiality as central qualities of the target setting, i.e., system that support interactions that include enacted and embodied performances in shared physical spaces. A basic assumption is that a final interactive setting that requires enacted and embodied performances shapes the nature of the programming activity to hold a rather different character from when other forms of interactions are designed.

After a short overview of the perspective of embodied performances in computer programming, we present three brief cases that illustrate how developers manage the situation of building systems for different kinds of physical and performance-based interactions. The examples are used to emphasize a general shift in how to understand programming practice, and how it gets shaped by the variety of ways that emerge through users' interaction with digital technology that increasingly become physical and spatial in character. We end by discussing how an increased acknowledgement of physical aspects of programming suggests new challenges for the broader development of new resources for software development.

2. Embodied performance in computer programming

Performance in interaction design can be investigated through a range of different perspectives. One is the classic view of Brenda Laurel's Computers as Theatre (Laurel, 1993), where the computer program itself is looked upon as a "live performance", experienced in use by its users. The role of the developer is then characterised as a theatre director orchestrating the acts to be performed by the computer program. A second view is through the performance of users, who together and with the system act in a social setting, towards a real or imagined "audience" (e.g., Grint and Woolgar, 1997). A third perspective concerns the communicative acts of designers and programmers "in the making". Examples of methods that explicitly address such matters include test-running of low-fidelity prototypes through collaborative role-plays (Rettig, 1994) and methods of bodystorming (Oulasvirta et al., 2003), where designers explore a given context of use through acting out everyday activities in the environment of their target users. Recently, such bodily aspects have been increasingly brought into the conceptual discussions of interaction design at large, especially in the design of physical and mixed-media interactive environments (Jacucci, 2004; Ciolfi and Bannon, 2007). A related aspect, and which we will focus on here, concerns the general development of new technical solutions that extend the qualities of the resulting systems to allow for richer forms of manipulation, perception and technology use. An important aspect of such technologies is that they invite users, and also the developers, to engage in more physical, social, and bodily forms of interaction.

Fig. 1. Interaction designers engaged in a modelling activity.

To further understand how the concept of performance can be understood in these respects, we have chosen to explicitly draw attention to the activity of computer programming. We find this relevant as this activity has traditionally been viewed as a primarily intellectual activity – a view that has recently been challenged by a range of empirical studies (Button and Sharrock, 1995; Downey, 1998; Sharp, 2004; Chong and Hurlbutt, 2007). A view that puts its primary focus on the intellectual aspects of activity has also been substantially criticised in social and cognitive science in general, emphasising the socially and physically situated nature of creative work in professional practices (see, e.g., Schön, 1983; Suchman, 1987, 2007; Lave, 1988; Schatzki et al., 2001).

This perspective of studying programming as a socially and physically situated activity, strongly shaped by the character of the available resources, has elsewhere been referred to as *embodied programming*, drawing on Dourish's notion of embodied interaction (Fernaeus and Tholander, 2006b). This includes a general concern for bodily actions performed in order to interact with and through a programming environment, and also to the physical aspects of discussions, negotiations, and perception that goes on when people engage in construction of computational artefacts. Note however, that this is not to be mixed up with *embedded* programming, which concerns more specifically the sensing, behaviour, and control of physical appliances.

In the field of HCI, much work has attempted to develop ways of bringing social and material aspects of use settings to the centre of concern of interaction design practices, e.g., through contextual inquiry, participatory design methods, technology probes, and 'quick and dirty' ethnography. Bridging the gap between use and design settings is, however, not only a question of providing information about users and use settings to the designers. A recent focus on physicality in programming can be found for example in the motivations for pair programming, where two developers work together in front of a shared screen. In this way, errors are more easily detected, and problems are immediately discussed and addressed collaboratively. Other physical aspects noted in ethnographic accounts of extreme programming practice include the importance of a common workspace, so that participants more easily can create a shared understanding of what everybody in the team is currently engaged in. Instead of storing information about plans, goals, requirements, etc. in digital files, information is made publicly available as physical representations in the forms of phys-

Please cite this article in press as: Jonsson, M. et al., Setting the stage – Embodied and spatial dimensions in emerging ..., Interact. Comput. (2008), doi:10.1016/j.intcom.2008.10.004

ical story cards and sticky notes on whiteboards (e.g., MacKenzie and Monk, 2004; Sharp, 2004; Chong and Hurlbutt, 2007; Martin et al., 2007).

In this paper, we discuss how experiential properties of specific end-user settings are addressed concretely by programmers in their practice. This could be seen as the reverse problem as that explored in *palpable computing* (Büscher et al., 2007), which is concerned with making the "invisible" internal aspects of a computational system, e.g., system architecture and software components, easier for end-users to talk about and relate to in participatory design practices. Similarly, tangible programming projects are commonly addressing how physical representational forms may make abstract programming concepts more concrete. Here instead, focus is on how the physical end-user context is dealt with in conventional, PC and text-based programming settings. The intention is to start identifying some of the key challenges in designing and realising systems for physical action and bodily performance.

3. Three cases of programming for embodied performance

This section provides three example cases of designing for interactive settings where embodied performance is a central property, and how this influences the activity of programming. The aim is to explore the multitude of use settings that developers may need to relate to. Rather than documenting the practices in detail, we have chosen to describe specific themes of the three different settings, pointing at how they illuminate certain aspects and challenges that programmers have to face when designing for such settings. Through these cases we attempt to contribute to an understanding of some of the dimensions involved in design of technologies involving physical experiences, and in particular, the interplay between programming practice and its resources, the physical contexts, and the material properties of the interactive devices.

First, we examine a golf simulator setting, which is a case where the target activity is heavily based on physical action and full body interaction. Our second case concerns the design and control of interactive performances in public spaces, in the form of live media-rich presentations at large exhibitions. Finally, we examine the activity of programming a certain type of autonomous embodied interactive devices called GlowBots. An overview of the three cases is provided in Table 1.

Each setting was studied through a light-weight ethnographic study, based on observations and interviews with programmers at their respective work places. After explaining our research interests in their work, we asked the developers to perform and explain typical activities in their programming practice, and especially how they dealt with the very physical aspects of the interactive technologies that they developed.

Table 1
Characteristics of the three cases.

Use setting	Programming setting
Golf simulator	
Full-body interaction including non-computational artefacts as "interface" elements, i.e., real golf clubs and balls, and physical space	PC-based interaction in conjunction with a physical stage for debugging and fine-tuning the simulator
Multimedia show	
System tailored for a specific location, interaction includes a complex collaborative interplay between people on and behind a physical stage	Programming activity is intertwined with the creation of a physical environment, as well as with the final end-user activity setting
GlowBots	
Handheld physical interaction, where multiple devices in combination work to frame the larger social setting, e.g., to turn off the lights and perform physically with the displays	Programming and testing happens on very different devices, emphasising loops of testing, debugging, and tuning the interactive properties

3.1. Designing for full body interaction

A range of interactive technologies requires users to engage in extensive bodily engagement to interact with a computational system. Such systems include sports arcade games, simulators, and training applications for different settings. Physical golf simulators is an example of this class of systems, allowing people to practice their hobby by hitting real golf shots using real clubs and balls into a screen projection of a golf course. Naturally, building such a system will take on a different character than the more commonly observed cases of developing systems for PC and laptop-based modes of interaction. To explore these aspects further, we visited a company that focuses on building such golf simulators.

The observed setting is situated in a golf training centre consisting of a number of small 'booths' each containing an interactive golf simulator environment that are used by individuals or small groups of players. Each booth consist of a large wall projection of a 3D-rendering of a golf course in combination with a sensor-based device for capturing the physical properties of the golf shot, such as ball speed, direction, and spin, to calculate the trajectory and indicate where in the simulated environment the ball will end up (Fig. 2).

The setup of the system provides an interactive context that involves two distinctly separate modes of interaction, first, hitting golf shots to actually play the game, and second, interacting with the simulation to carry out the actions necessary to play in the manner one wants. This involves indicating to the system what club one is using, aiming the screen projection in the direction one wants to hit the shot, perceiving information presented on the screen regarding distance to the hole, speed, and direction of the wind, elevation, etc. This requires designers to support the users' possibility of shifting orientation between viewing the screen projection as a golf course and viewing the information presented to control the progression of the simulation. The first being a highly physical mode of interaction while the second being more of an information seeking mode of interaction.

Collaborative and social dimensions are important both to how golf is generally played in real-life settings, but was also prominent in our observations of how the game is played in the golf simulation centre where it was set up. Addressing these dimensions requires the developers to consider how the physical and spatial setting supports offline aspects of the interaction around the system such as observing other's shots, commenting on them, comparing them to those of oneself. This is supported both by how the spatial and physical configuration of the simulation cater for watching by audience and co-players. It is also addressed in the simulation technology by leaving traces of the trajectories of the shots of the other golfers that remain on the screen projection. In our observations of the developers building the system they use a conventional PC-based programming environment to control the simulation technology. Their golf simulation technology is based on models and principles from virtual game-worlds used in traditional desktop settings. A key challenge in this development is hence to use a traditional programming environment in building a technology that involves a highly physical activity conducted in a constrained physical space.

Programming and testing in this environment naturally involves addressing a number of issues in which physical and spatial qualities become salient. As non-computational physical objects such as club and ball are central interactive resources testing,

Please cite this article in press as: Jonsson, M. et al., Setting the stage – Embodied and spatial dimensions in emerging ..., Interact. Comput. (2008), doi:10.1016/j.intcom.2008.10.004

Fig. 2. An illustration of the physical use setting of the golf simulator and the two different modes of interaction with the system while playing, as well as when programming and debugging. The left picture shows interaction using golf ball and clubs, and the right shows the PC with which the users have to interact to for example select course and clubs.

e.g., the correctness of the physical measurements cannot be simulated on the PC only. Testing and debugging thereby requires the programmer to use a physical setup of the system and hit actual golf shots to generate different test values. The opposite of such technical testing procedures involves addressing how the system supports the imagined experience and the appropriate feeling of actually hitting real golf shots on a golf course. A central issue here regards addressing how users will experience the spatial qualities of the setup so that it gives a realistic sense of how the course projection is perceived. Achieving this requires extensive fine-tuning of the specific placing of the sensors and the position of the hitting-mat hit in relation to the room and the screen projection.

3.2. Programming for performance in large public spaces

Performances in public spaces have been a popular domain for exploring novel interaction forms and technologies. Such settings include musical performances (Jordà et al., 2007), place-specific artworks (e.g., Coffin, 2008) and one-off pervasive game experiences (Jonsson et al., 2006). One kind of system intended for such a public interactive space that we have chosen to investigate here is multimedia presentations at larger exhibitions. The activity of setting up and controlling such systems provides an interesting case to examine due to the various performative, embodied and spatial factors affecting the activity.

The systems specialised on at the second development company that we visited, are aimed for settings that typically include a stage where one or several presenters give speeches accompanied by visual multimedia material displayed on large displays (see Fig. 3). The multimedia material can be composed of static text/image slides in combination with video streams and/or computer animations. An important difference to other multimedia presentations is that this material is spread out over several screens, is often combined with live video footage of the presenter, and typically also includes a light show with directed and ambient light accompanying the presentation as well as spatial audio accompanying the various video streams. There are also several points of interaction both with the system and between the different people involved. First, the show is performed collaboratively by the people on stage and a number of people (who are also the designers/programmers of the presentation) located in a backstage control room. The presenter has control over what slides and video that are currently being displayed, either directly or indirectly by subtle cues to the control room staff. Except for interacting with the presenter the control room staff also has to coordinate their actions between them. At certain types of events the presentation is also combined with interactive features to, e.g., allow for voting or other feedback from the audience, and in other ways of engaging the audience in the activity.

The presentation activity can be divided into two phases, the setup (programming) phase, and the performance (execution) phase. The setup phase focuses on the creation of the stage, both in a literal physical sense, but also in a more general sense as to create a framing of the performance that will take place. The

Fig. 3. Physical setting of a multi-screen multimedia presentation accompanying an on-stage presentation (left), and the control room used to set up and control the various elements in the presentation during the show (right).

Please cite this article in press as: Jonsson, M. et al., Setting the stage – Embodied and spatial dimensions in emerging ..., Interact. Comput. (2008), doi:10.1016/j.intcom.2008.10.004

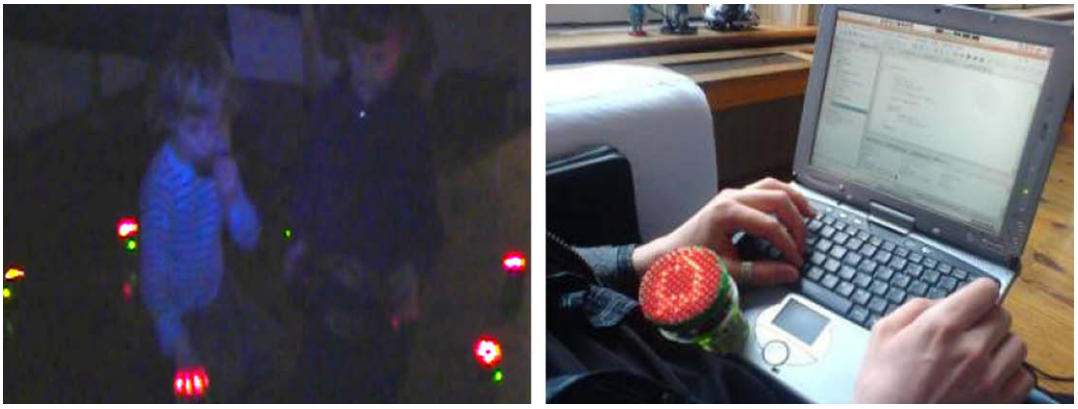

Fig. 4. Children interacting with the GlowBots, with the light in the room switched off and the glowing devices spread in a circle across the floor (left). Physical setting of programming a GlowBot (right).

physical setting, including stage, projection surfaces, etc., is to various degrees tailored for the actual event and is sometimes designed by architects. During the setup phase, the control room staff has to consider a number of factors related to the specific physical situation. The presentation material will have to be adapted to fit the specific setting, including, for example, the adjustment of video streams to fit oddly shaped or curved projection surfaces.

Staging the interactive setting includes to design both the interaction with the system as well as the interaction between the different actors in the setting. The control room staff have the dual role of being both the designers of the specific interactive setting as well as key performers in the actual execution of the show. Similar to reported experiences of building pervasive game experiences (Crabtree et al., 2007), this orchestration work means that behind the scenes staff become co-producers of the activity and that the end-user interaction is inseparably intertwined with the development process.

The developers will also have to take into consideration how the coordination of the presentation between presenter and control room staff should be established. Thus, when creating the setting they create a build-once run-once environment consisting of both physical and computational material that is tailored to a specific place and a specific collaborative interactive activity.

3.3. Programming for performance with embodied interactive devices

An increasing number of tangible interactive artefacts include sensor technology used for highly specific purposes such as interactive toys, digital cameras, robotic, and mobile devices. Such devices generally include stand-alone software executed within the device itself rather than working as an interface towards a more generally usable computer system.

One kind of system in this category, GlowBots, takes the form of a set of mug sized interactive robots that show visual patterns on a round LED-display. Users interact with these either by moving them around on the surface or by gently shaking them in different directions. As two robots stand next to each other, they may communicate directly for instance by exchanging patterns on their visual displays. Important features of the GlowBots are first of all their visual glowing display and their ability to move autonomously, and also that they are interacted with through shake-based physical manipulation. Thus, the way they are interacted with is rather unique, making this development situation similar to a range of other devices, as there is no programming environment that specifically addresses this particular form of interaction.

Compared to the other two cases, the GlowBots are less bound to any specific physical location. This does, however, not mean that the physical setting is insignificant with respect to how the GlowBots are being put to use. In observations of children playing with the GlowBots, the children were actively incorporating the robots into the physical environment, adapting it by, for example, turning off the lights and rearranging furniture. The room with the interactive devices thus became a stage for interaction that was constantly being recreated and reshaped by the activities taking place there (see Fig. 4).

The GlowBots are programmed using a conventional PC-based programming environment without any particular features for interpreting of the kind of sensor data generated by the GlowBots. The physically specific nature of the interaction also means that such sensor data is very difficult to simulate or predict. The programming activity is typically an extensively iterative process where the code is written and compiled on the PC, and then uploaded to one or a few GlowBots. As there is no proper way of simulating the GlowBots behaviour on the PC any new or altered behaviours will have to be tested by physically running the new software on the GlowBots.

One of the most important and time consuming aspects of programming the GlowBots in this way concerns not so much the generation of new program behaviours as of testing and debugging the details of the behaviour through fine-tuning various properties and parameters. Each time a parameter is to be reset, the whole code needs to be recompiled and uploaded to the physical device. The very specificity in the details of interacting with, i.e., the specific accelerometer and how that will affect the diodes on the screen display means that it would be impossible to do this form of debugging in a 'simulator mode' with an emulator on the PC only. Together with the two examples above, this exemplifies the shift towards physicality in programming also for debugging activity. While debugging is an important part in all programming activity, we here see how it requires a particular focus on embodied and experiential aspects of the system being developed, rather than on particular algorithmic structures.

4. Discussion

Embodied performance with computing devices is a significant aspect of the current trend towards pervasive, mobile, media-rich, and socially oriented technology. The various approaches in the three cases that we have presented above were intended to illustrate an expanded perspective of contemporary interaction design and programming activities, and also to put focus on some of the issues that have to be considered when designing tools to support

Please cite this article in press as: Jonsson, M. et al., Setting the stage – Embodied and spatial dimensions in emerging ..., Interact. Comput. (2008), doi:10.1016/j.intcom.2008.10.004

these practices. Each case emphasises some challenges that seem to become increasingly important when programming for this type of embodied and performance-based settings.

4.1. The role of physical space

The first case shows how building a physical golf simulator emphasises not only a cognitive shift between the activities of programming and debugging, but also a shift in physical engagement. The physical and spatial properties of the target interaction setting thereby became central aspects of the actual programming practice. The designers and programmers did not only have to consider physicality as important in the target setting, they also had to explicitly draw in place-specific properties into their own activity. Similarly, the developers in our second case had to explicitly address the character of the exhibition hall, physical properties and locations of screens and other devices, into their designs efforts. Thereby, they are not only programming for a specific place, the system they are building actually reconstruct the place through the interaction and embodied performances that it affords. Even if customization and adjustment to place-specific properties is common within different areas of software systems design, the cases described here differs in the explicit focus on spatiality and the physical embodiment of the resulting system.

The specifics of physical space get further emphasised as the programming environments in all the cases presented are dominated by traditional PC-based tools for individual use, while the target interaction settings are dominated by physical action and performances in larger social spaces. This emphasises a gap between the coding, debugging and performance as separate activities. Shifting between the different modalities requires significant creative work on behalf of the programmer, as illustrated by the golf simulation programmer having to physically hit a golf shot in order to generate test data, or the intense fine-tuning of the behaviours of the GlowBots. These observations are supposedly also valid in several other development settings as it highlights how the dominant modalities of most development tools often differ significantly from the target interaction settings. For applications that rely extensively on experiences of physical, spatial, media-rich, and social interactions we argue that these issues are particularly prominent.

4.2. Designing stages for embodied performance

The second case showed a build-once run-once scenario, where the fine-tuning and control of the run-time activity is turned into an act of in situ performance by the programmers in the form of control room workers. Programming for such settings thereby inevitably incorporates a range of physical and spatial constraints that have immediate consequences for how the interaction will take form, putting emphasis on the final 'computer system' as providing only for a fraction of the whole interactive experience.

Even though only a subset of the physical actions of the users are actually sensed by the systems involved, they have significant influence on how the end-users will experience their interaction. This is most evidently illustrated by the way that the developers of the exhibition presentations have to program for situations in which a controller can adapt the execution in run time, but is present also in the other cases. All three cases target interaction settings where collaborative and social action is a key dimension. Even though the systems' main features were not to support collaboration between the participants these are features that drive the interaction. This makes it difficult and supposedly undesired for the programmers to control exactly how the resulting application will be taken into use in each specific social context. In line with the arguments proposed by Sengers and Gaver, (2006), the programmers in all three cases have to develop a flexibility in their stance towards their target use settings in order to effectively support the kind of interaction they intend, but without attempting to control every aspect of it. This includes the shifts in interaction modes in the golf simulation case, the different perspectives of the users of the exhibition presentations, as well as the open-ended character of interacting with the GlowBots.

Rather than viewing the intended interaction through one particular user, a number of potential actors may choose to attend to the output from the systems from their specific viewpoints and interests. Programmers then face a challenge concerned with adapting to the social and bodily actions that different user roles such as golfer, presenter, and audience may take with the design. This can be illustrated through the social interactions inherent in the golf simulation and the multitude of perspectives between presenter and the audience in the exhibition presentations. A consequence of this is that our understanding of programming code should not be framed primarily as representations and execution of algorithmic structures, but rather as providing new resources for action – in this case this includes resources for programming action, as well as for social and bodily actions in the interaction with the technology being developed. More generally, this suggests a shift in programming practice from taking properties of the interactive system as the primary unit of analysis, to instead primarily examine the interactive system in relation to its physical, spatial, and social context of use.

4.3. Device-specific qualities of interaction

The specific characteristics of the target devices play an important role in all three cases. The golf simulation case required a novel setup of sensors in a physical space, the second case required adjustment to unique hardware configurations for each show, and the GlowBot example emphasised the explorative and experiential nature of programming, debugging, and tweaking. In all cases, the gap between the largely mouse-and-keyboard based programming environments and the experiential properties of interacting with the target setting became evident.

Considerations of device-specific properties of interactive artefacts could here be interpreted as quite recent aspects of software development. Computer programming has, however, always involved addressing the physical nature of target devices, with the overall goal to generate instructions that can be interpreted in hardware (see, e.g., early accounts of copying machine development in Suchman, 1987; Sharrock and Anderson, 1994). Programming projects, also for more conventional use settings, nearly always need to address aspects of how the target system should be physically controlled and experienced, e.g., via mouse, keyboard, touch screens, etc. However, what we find significant is that these aspects here become recognised as primary aspects in the activity, of equal importance as the actual code that the work results in.

One of the reasons for the device-specific properties to become especially salient in these cases could be traced to the different modalities of the programming and the end-user setting. The cases that we explored here were all based on text-based programming environments, yet the activities were strongly shaped by the kind of physical actions that end-users should perform. A critical issue that was repeatedly dealt with was to bridge between the different modalities of the physical and social settings of programming and use of the system, which significantly shaped the approaches taken in the development projects. One way of addressing this issue could be to design resources and representations for programming with properties that in a more direct sense support developers in anticipating the physical and spatial qualities of the resulting interactive system, e.g., by developing different kinds of tangible pro-

Please cite this article in press as: Jonsson, M. et al., Setting the stage – Embodied and spatial dimensions in emerging ..., Interact. Comput. (2008), doi:10.1016/j.intcom.2008.10.004

gramming resources (Edge and Blackwell, 2006). We see this as a promising approach for addressing the cumbersome shifts between the different modalities that repeatedly occurred in the observed practices.

The three cases, with their different physical characteristics thereby emphasise that the computational media that programmers work with could never be considered as completely immaterial and disembodied, but rather a material that is always concretely situated in the shared world in which we live and act. An inherent aspect of software development is then to find bridges between what one wants to express in the computational media, and how this will take effect concretely in the resulting system behaviour in a specific setting. With the growing range of pervasive, ubiquitous, and physical interactive appliances around us, device-specific aspects of materiality, physicality, and spatiality become increasingly important to address by those who intend to build these systems. Support for such considerations requires further investigations into the conditions and requirements of these new interactive settings, as well as new development of toolkits specifically designed for such work.

5. Conclusions

Aspects related to physicality in the design of artefacts and environments have always had to be addressed in practices like industrial design and architecture. What is new is that the physical appliances and environments are becoming increasingly more computerized and thus to a larger extent becomes a target domain for computer programming. It then becomes interesting to examine how existing programming practices are challenged by the specific characteristics of this domain. What has been shown here is that these new settings have spatial and physical properties that make them substantially different from PC-based settings, and those programmers need to bring these aspects into the core of their practices of building, testing, and debugging. The multitude of approaches in the presented cases reveals an expanded perspective of what constitutes a programming activity, emphasising the interlinking between design, programming and performance with the system that is being built.

In our analysis of the different cases, some issues became especially prominent. The first observation was that the activity of designing and programming interactive systems *cannot be entirely distinguishable from the physical setting* where the final system is to be used and how programmers actively have to adapt their practices to the unique characteristics of the specific setting in order to debug and fine tune the interaction. The second observation concerned the *interlinking of programming and performance* with the completed systems, and the importance of acknowledging that the interactive setting is larger than the computational system, and thus that designers adapt their practice to also target interaction within the setting that is not directed only towards the system. A last observation concerned the importance of considering programming for *device-specific qualities of interaction*. This especially concerned the gap between the immaterial characteristic of text-based programming and the resulting highly physical use settings, evident, for instance, in the shifts between coding and testing. This suggests that further effort needs to be placed into understanding how new tools could be designed to support programmers of systems for extensive physical and bodily engagement.

In the three cases presented here, the specific interactive qualities strived for in each individual project are very different in kind. Similar to most interaction design work, every development project has unique qualities and requirements and it would be problematic to propose a single approach to programming these kinds of environments. Consequently, we will not attempt to point out any specific implications or methods for programming of embodied performances since each new system requires its own creative process for specifying the functionality of the system itself. However, a focused reflection over a range of different cases contributes to further our understanding on how to design and set up the physical and social resources for developers in their work towards designing, programming and fine-tuning the experiential properties of the systems that they are building. Here, we have attempted to contribute to an understanding of some of the dimensions involved in design of technologies involving physical experiences, with emphasis on the close intertwining between programming practice and its resources, the physical contexts, and the material properties of the interactive devices. Unpacking these layers of physicality is something that programmers deal with practically within their daily routines of building interactive systems, but also something that HCI research need to address in order to further our understandings of people and contemporary technology.

Acknowledgements

We thank the golfing simulation company Simway AB, the media technology and event design company Primetec AB, and Mattias Jacobsson, project leader of the GlowBots research project, for sharing their experiences of programming, development, and interaction efforts with us.

References

Button, G., Sharrock, W., 1995. The mundane work of writing and reading computer code. In: Have, P.t., Psathas, G. (Eds.), Situated Order: Studies in the Social Organization of Talk and Embodied Activities. The Press of America, Washington, DC, pp. 231–258).

Büscher, M., Christensen, M., Hansen, K.M., Mogensen, P., Shapiro, D., 2007. Bottom-up, top-down? Connecting software architecture design with use. In: Hartswood, M., Ho, K., Procter, R., Rouncefield, M., Slack, R., Büscher, M. (Eds.), Configuring User–Designer Relations: Interdisciplinary Perspectives. Springer-Verlag, Berlin.

Canfield Smith, D., Cypher, A., Tesler, L., 2001. Novice programming comes of age your wish is my command. In: Lieberman, H. (Ed.), Programming by Example. Morgan Kaufmann Publishers, San Francisco, CA, pp. 7–19.

Ciolfi, L., Bannon, L.J., 2007. Designing Hybrid Places: merging interaction design, ubiquitous technologies and geographies of the museum space. Co-Design 3 (3), 159–180.

Chong, J., Hurlbutt, T., 2007. The social dynamics of pair programming. In: ICSE 2007, Minneapolis, USA.

Coffin, J., 2008. Robotany and Lichtung: a contribution to phenomenological dialogue. In: Proceedings of the 2nd International Conference on Tangible and Embedded Interaction. ACM, Bonn, Germany.

Crabtree, A., Benford, S., Capra, M., Flintham, M., Drozd, A., Tandavanitj, N., Adams, M., Row Farr, J., 2007. The cooperative work of gaming: orchestrating a mobile SMS game. Computer Supported Cooperative Work 16 (1–2), 167–198.

Dourish, P., 2001. Where the Action Is: The Foundations of Embodied Interaction. Massachusetts Institute of Technology, Cambridge.

Downey, G., 1998. The Machine in Me: An Anthropologist Sits Among Computer Engineers, Routledge.

Edge, D., Blackwell, A.F., 2006. Correlates of the cognitive dimensions for tangible user interface. Journal of Visual Languages and Computing 17 (4), 366–394.

Fernaeus, Y., Tholander, J., 2006a. Designing for programming as joint performances among groups of children. Interacting with Computers 18 (5), 1012–1031.

Fernaeus, Y., Tholander, J., 2006. Finding design qualities in a tangible programming space. In: Proceedings of CHI 2006, ACM press, Montreal, Canada.

Frei, P., Su, V., Mikhak, B., Ishii, H., 2000. Curlybot: designing a new class of computational toys. In: SIGCHI Conference on Human Factors in Computing Systems. ACM press, The Hague, The Netherlands.

Grint, K., Woolgar, S., 1997. The Machine at Work. Polity Press.

Hartmann, B., Abdulla, L., Mittal, M., Scott, R.K., 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In: Proceedings of the SIGCHI Conference on Human factors in computing systems. ACM, San Jose, CA, USA.

Heath, C., Luff, P., 2000. Technology in Action. Cambridge University Press, Cambridge.

Horn, M., Jacob, R.J.K., 2007. Tangible programming in the classroom with tern. In: CHI2 007, ACM Press, San Jose, USA.

Hornecker, E., Buur, J., 2006. Getting a grip on tangible interaction: a framework on physical space and social interaction. In: Proceedings of CHI 2006, ACM Press, Montreal, Canada.

Please cite this article in press as: Jonsson, M. et al., Setting the stage – Embodied and spatial dimensions in emerging ..., Interact. Comput. (2008), doi:10.1016/j.intcom.2008.10.004

Jacucci, G., 2004. Interaction as performance Cases of configuring physical interfaces in mixed media. Faculty of Science Department of Information Processing Science. University of Oulu, Oulu, Finland.

Jonsson, S., Montola, M., Waern, A., Ericsson, M., 2006. Prosopopeia: experiences from a Pervasive Larp. In: International Conference on Advances in Computer Entertainment Technology. ACM, Hollywood, CA, USA.

Jordà, S., Geiger, G., Alonso, M., Kaltenbrunner, M., 2007. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In: Proceedings of TEI 2007. ACM Press, Baton Rouge, Lousiana, USA.

Kahn, K., 1996. ToonTalk – an animated programming environment for children. Journal of Visual Languages and Computing 7 (2), 197–217.

Klemmer, S.R., Hartmann, B., Takayama, L., 2006. How bodies matter: five themes for interaction design. In: Proceedings of DIS '06. ACM Press, Penn State, PA, USA.

Laurel, B., 1993. Computers as Theatre. Addison-Wesley.

Lave, J., 1988. Cognition in Practice. Mind, Mathematics and Culture in Everyday Life. Cambridge University Press, Cambridge.

MacKenzie, A., Monk, S., 2004. From cards to code: how extreme programming re-embodies programming as a collective practice. Computer Supported Cooperative Work (CSCW) 13 (1), 91–117.

Martin D., Rooksby J., Rouncefield, M., 2007. Users as contextual features of software product development and testing. In: Proceedings of the 2007 International ACM Conference on Supporting Group Work (GROUP), 4–7 November. Sanibel Island, FL, USA, pp. 301–310.

McNerny, T.S., 2004. From turtles to Tangible Programming Bricks: explorations in physical language design. Personal and Ubiquitous Computing 8, 326–337.

Oulasvirta, A., Kurvinen, E., Kankainen, T., 2003. Understanding contexts by being there: case studies in bodystorming. Personal and Ubiquitous Computing 7 (2), 125–134.

Raffle, H.S., Parkes, A.J., Ishii, H., 2004. Topobo: a constructive assembly system with kinetic memory. In: SIGCHI Conference on Human Factors in Computing Systems, ACM Press, Vienna, Austria.

Rettig, M., 1994. Prototyping for tiny fingers. Communications of the ACM 37, 21–27.

Schatzki, T., Knorr-Cetina, K.D., Savigny, E. v. (Eds.), 2001. The Practice Turn in Contemporary Theory. Routledge, London.

Schön, D.A., 1983. The Reflective Practitioner: How Professionals Think in Action. Basic Books.

Sengers, P., Gaver, W., 2006. Staying Open to interpretation: Engaging multiple meanings in design and evaluation. In: Proceedings of DIS 2006, ACM Press, Penn State, PA, USA.

Sharp, H.a.R., 2004. An ethnographic study of XP practices. Empirical Software Engineering 9 (4), 353–375.

Sharrock, W., Anderson, B., 1994. The user as a scenic feature of design space. Design Studies 15 (1), 5–18.

Suchman, L., 1987. Plans and Situated Actions. Cambridge University Press, Cambridge.

Suchman, L., 2007. Human–Machine Reconfigurations, second ed. Plans and Situated Actions Cambridge University Press, Cambridge.

Tholander, J., Karlgren, K., Ramberg, R., 2008. Where all the interaction is: sketching in interaction design as an embodied practice. In: DIS'08.

Please cite this article in press as: Jonsson, M. et al., Setting the stage – Embodied and spatial dimensions in emerging ..., Interact. Comput. (2008), doi:10.1016/j.intcom.2008.10.004